ASHTEX : An interactive previewer for TEX or the marvellous world of ASHTEX

Laurence Gallot-Rideau

INRIA

CENTRE
SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.(3)954 90 20

Rapports Techniques

N° 75

ASHTEX: AN INTERACTIVE PREVIEWER FOR TEX OR THE MARVELLOUS WORLD OF ASHTEX

Laurence GALLOT-RIDEAU

Septembre 1986

Ashtex: Un Programme Interactif d'Epreuvage pour TEX

Ashtex: An Interactive Previewer for TEX or The Marvellous World of Ashtex

Résumé

Dans ce rapport nous présentons un outil multi-fenêtres interactif, permettant de visualiser sur un écran bitmap des documents formattés par TEX. Cet outil a été implémenté à l'aide d'un gestionnaire d'écran appelé ASH. Ce gestionnaire d'écran fait partie d'un Environnement de Poste de Travail développé à l'Université de Brown: BWE (Brown Workstation Environment). A l'heure actuelle, Ashtex tourne sur SM90 et SPS7 avec écran bitmap Numelec, sur SPS9 et sur SUN.

Abstract

This paper describes an interactive multi-window previewer for TEX. This tool is built on a screen handler called ASH, that is part of Brown Workstation Environment developed at Brown University. Today, Ashtex runs on SM90 with the Numelec Bitmap screen and on SPS9, and on SUN workstations.

Ashtex: An Interactive Previewer for TEX or The Marvellous World of Ashtex

Laurence GALLOT-RIDEAU
INRIA SOPHIA-ANTIPOLIS
Route des Lucioles
06560 Valbonne, France

1. Introduction

Using TEX for typesetting all our documents, we have required tools to help us in this work. We have adapted to our computers the Brown Workstation Environment, BWE, which makes it possible to design graphical applications easily. With BWE we have realized an interactive multi-window previewer for TEX that we present in this paper.

We first explain what a previewer is and why we need such a tool for displaying documents typeset with TEX. Then we describe the specific functionalities that are desirable. Next we give the details of the implementation of our previewer, that we have named Ashtex, and finally we describe the problems we have encountered, and how they were solved when possible.

2. Description of a Previewer

2.1. What is a previewer and why does a TEX user need such a tool?

When you prepare a document using a typesetting system, such as TEX, the initial work consists in typing a correct source file on a standard alphanumeric terminal. Next the TEX program is run on this file to obtain an intermediate file called a *dvi file*, and finally by interpreting this dvi file with a driver output is generated. Notice that the driver used depends on the type of the output device.

While the source file consists of the text and of command strings for the typesetting, one doesn't see very precisely what the final document will look like. Thus very quickly, one wants to see output. The most usual and simplest way of looking at the result consists of outputting the document on paper. Unfortunately this method is often slow and may be expensive. For instance, there is often only one printer for several workstations; to use the printer files must be transfered to the appropriate machine and then wait for the output. If there are many users for the printer, this may take awhile. Futhermore pages on such a (laser or electrostatic) printer are very expensive — we have computed that a page on our laser printer is about one Franc, which is at least three times more expensive than photocopying. It is expensive and cumbersome to output 10 draft copies before getting the output just right.

One often wants to look only at a part of a page. For example, for designing complicated equations or arrays, it may take many tries before the output is correct, or sometimes the page layout is to be checked. While it is not reasonable to get hardcopy output for each modification on the source file, it would be nice to be able to look at the document quickly without outputting it on paper.

Since personal workstations with high resolution display have recently become widely available, one would like to view the TEX output directly on the screen. This method makes the preparation of a document easier and less expensive. A tool that enables the user to look at his document without outputting it on paper is called a previewer. A previewer allows the user to look at *what he will get* on the paper. The layout on the screen is exactly the same as on paper. The only difference comes from the resolution of the display device, and the result on the screen may look less beautiful than the hardcopy version. Although the lower resolution of the display (around 100 dots per inch for the Numelec display against 300 dots per inch or more for a laser printer) results in the characters being less beautiful, it seems very sufficient for preparing a document. The user can quickly see the results and make modifications.

2.2. What does the user want from a previewer?

Clearly the user wants to look at any page in the document and does not necessarily want to view the pages in sequential order. However, when viewing a page it is often the case that the next or previous page of the document is the one the user wants to look at next. Thus the previewer should provide facilities to get the next or the previous page easily while still providing the freedom to see any arbitrary page. Also he may want to look at more than one page at the same time on the screen — for side-by-side comparison. Perhaps the pages to be compared even come from different documents. Thus, the previewer must handle requests for multiple pages and multiple dvi files.

Depending on the output device and the size of the pages in the user's document, it may not be possible to fit the whole page on the bitmap screen at once. Thus, the user must be able to "scroll" through the page, both in the horizontal and vertical directions. That is, if the page is too tall and cannot appear totally on the screen, the user needs to be able to move the page up or down quickly to see a different portion of the page.

A multi-window system makes it possible to realize these needs. Each page viewed is in a window (a portion of the screen delimited by a border). If there are several windows (pages) on the screen, they may overlap one another. If the multi-window system used is powerful enough, the pages can be moved around. If there are many pages (windows) on the screen, it is necessary to quickly move the page to be viewed to the forefront so that it is no longer obscured by the others. It should be possible to change the size of a window: if one wants to keep visible only a small part of a page, one can resize the corresponding window and move it in a corner of the screen where it remains visible. The user can also choose the number of pages that he wants to see on the screen at the same time. For example, he may choose to only have one or two windows; pages to be viewed are put in one of these windows. However, he may have a window on the screen for each page he views — perhaps a bit cluttered for a 15-page paper. But the user has control of creating and destroying windows.

3. Implementation of Ashtex

We now examine how we have implemented the previewer facilities of the previous section. To realize our previewer we needed a multi-window system. We used the Brown Workstation Environment

(BWE) that we have adapted for use on the SM90 workstation with Numelec bitmap screen and on Bull SPS9.

3.1. Presentation of BWE

BWE is a workstation environment developed at Brown University, for the design of interactive graphical applications. This environment consists of different libraries, and provides facilities for using powerful hardware components such as high-resolution bitmapped displays and locator devices (e.g. a mouse) with a minimum amount of effort. In particular, it allows input management (using both the standard keyboard and a locator device), sophisticated graphical output to multiple overlapping windows, and text and graphics editing.

The main feature of this system is its portability; it can be adapted to machines running a Unix system and including a high-resolution output device (monochrome or color) and any locator device (mouse, light pen, data tablet, etc.) Portability is obtained by coding in the C language for Unix and by carefully isolating all the machine dependencies in virtual device interfaces for input and for output. Adapting BWE to a machine requires only writing these interfaces.

At Brown the system has been developed on Apollo workstations, and then adapted to the Sun workstation. At INRIA Sophia, we adapted it for SM90 workstation with the Numelec bitmap display and SMX system, and for SPS9 computer running Unix. Our objective here is not to describe in detail each component of the BWE system (for more information refer to the BWE documentation [1].) However we present here the ASH library which makes it possible to implement our previewer for TEX.

3.2. Presentation of ASH

ASH is a low-level screen handler completely independent of the machine and of the output device used. It allows the user to create, modify, and manipulate bitmaps and representations of these bitmaps on the display. The main data structure manipulated by ASH is the window. An ASH window is a virtual bitmap on which the application draws and outputs text. A window can be displayed on the screen in full, in part, or not at all.

The main part of the work of the screen handler ASH consists in computing the visible area associated with each window and displaying the border that delimits the visible part of each window. Windows are maintained by ASH in two ways. First, they are maintained in a tree-hierarchy as they are created. The screen is the window associated with the root of the tree-hierarchy, and each window is the child of the current window at the time when it was created. For each window, its visible part is entirely contained in the visible part of its parent. The second way ASH maintains the windows is to have for each a list of its children ordered for the display. The ordering of the children is such that any window above a given one may obscure this window, however, no window below it in the list may obscure any portion of the given window.

For drawing and outputting text, each window has its own set of graphic attributes: current color, text color, background color, combination rule (mode for combining bits: or, xor, etc.), fill pattern, line style, and font. These attributes can be changed at any moment during the execution of the application program.

3.3. Description of the output screen

The screen of an Ashtex session consists of windows for outputting the pages, menus, and a dialog window for the output of messages and keyboard input.

3.3.1. The output windows

As we have seen before, pages are almost never fully displayed on the screen because of their size. In fact, once a page as been computed (as an array of bits), it is stored in a virtual bitmap, and partly displayed on the screen in a window.

The windows we have implemented look like *MacIntosh* windows. They have a header, in which the dvi file's name and the number of the associated page are written. The user can move the window by clicking in the header. At this time a ghost (rectangular box) appears around the window and follows the moves of the mouse until the mouse button is released. The last position of the ghost indicates the new position of the window. In the left corner of the header there is a small black square that allows the window to be destroyed.

The windows have two scrollbars. One scrollbar is vertical, on the right border of the window, and it permits vertical scrolling of the corresponding page in the window. The second scrollbar is horizontal, on the bottom border of the window, and permits horizontal scrolling. The fact that the entire page exists somewhere in memory (in a virtual bitmap) makes it possible to scroll very quickly and makes the scrollbar very useful.

A scrollbar consists of a rectangular area, with a square button at each end. In each button there is an arrow indicating a scrolling direction, and clicking in one of these buttons allows for a step by step (incremental) scrolling. In a scrollbar, there is also a small rectangular area (called a thumb) representing the part of the page that is visible in the window; the size of the thumb is proportional to the visible part of the page and the position of the thumb in the scrollbar represents the position of this visible part within the total page. To jump to another part of the page, one clicks on the thumb and then positions it (actually an outline or "ghost" version of the thumb moves) within the scrollbar; releasing the mouse button results in moving the thumb over the ghost and the corresponding portion of the page appears in the window. Depressing a mouse button in the scrollbar but not on the thumb results in the thumb moving by a given step and the page scrolling accordingly until the button is released or the designated point lies within the thumb.

A small square area, in the right bottom corner of the windows allows the resizing of the windows. When a button of the mouse is depressed in this area, a ghost appears around the window and grows (or shrinks) following the moves of the mouse until the button is released. The last size of the ghost becomes the new size of the window. When the window is resized, the header, scrollbars, thumbs, and the visible part of the page are resized accordingly.

3.3.2. The menus

In Ashtex we use two kinds of menus: a permanent one and two pull-down menus. The pull-down menus are accessed through the permanent menu. The permanent menu has an horizontal format and lies in the top left corner of the screen; it has five buttons named *stop*, *refresh*, *new dvi*, *new frame*, and *new page*.

The first button is used for stopping the program. Click this button and a three-button pull-down menu appears. These give three choices *clear* (to stop and clear the screen), *leave* (to stop and leave the screen as it is), and *cancel* (to remain inside Ashtex). Note that when the *stop* button is chosen if the mouse button is released outside of the pull-down menu, the command is also cancelled.

The button named *refresh* is used to refresh the screen. This is useful, for example, to remove parasitic messages from the system or from another user.

The third button of the permanent menu, named *new dvi*, is used to load a new dvi file. When this button is chosen the program asks for the name of the dvi file to load, and for the number of the page that one wants to see first. If the user asks for loading the same dvi file (with the

same name), the program asks him if he really wants to reload the same file. Note that pages can only be selected from the current dvi file. Thus alternating between pages from different dvi files requires that the dvi file be loaded at each switch. This is because Ashtex uses a very simple driver for dvi files. Thus information is only maintained about the "current" dvi file. Switching between dvi files requires some start-up overhead to load the information from the dvi file and read the corresponding fonts.

The next button, named *new frame*, is used to get a new window on the screen. The user has to give a point on the screen, where the top left corner of the new window will appear, and the number of the page that he wants to see in this new window. By default, we have chosen to create two windows at the beginning of the program. The user can destroy one window if he desires, or he can create some windows (with the *new frame*) button if he wants to works with more than two windows. In this (default) setup the user would see each new page requested appear alternating in the two windows; thus the screen always has the last two pages asked for.

Finally, the last button, named *new page*, is used to change the current page. When this button is clicked, an other three-button pull-down menu appears. The content of the header of this menu is the name of the dvi file and the number of the current page. The three choices offered by the menu are: *next* (page following the current one), *previous* (the previous one), and *other* (to get any page from the document). Clicking the *other* button, the user is asked to give the number of the page he wants. If the chosen page is not already in a window on the screen, it is computed using the current dvi file and written in some window. The window whose contents is the requested page is moved to the top of the screen, and this page becomes the current one. This implies in particular that the number of this page will appear in the header of the pull-down menu, when calling *new page*.

3.3.3. The dialog window

The last component of the output screen of Ashtex is the dialog window. It is a rather small window that appears in the right top corner of the screen when keyboard input is needed. It is used for communicating with the user, for example, sending error messages or requesting the name of a dvi file or what page to show.

3.4. Computing and displaying the pages.

When a page that isn't already in a window on the screen is requested this page has to be computed and then copied in a window. The width of the window is determined by the width of the page in the dvi file, while the height is computed so as to be in the "golden ratio" with the width. Computing a page is obtained by using a driver that interprets the dvi file and generates an array of bits. For Ashtex we use the same driver as for our Canon printer. Because of the different resolutions of the printer and of the bitmap, the fonts used differ but the main part of the computation of a page is the same. When the called page is computed, the program has to choose which window of the screen will be associated with the new page. For that purpose we use a *virtual memory managing* technique: the chosen window is the least recently-used one. The program maintains a list of the windows displayed on the screen, when a window becomes the current window, it becomes the first element of the list and pushes the other windows behind it in the list; the window chosen to show a new computed page is the last of the list. If there is only one window on the screen, for each page computed, the contents of this window is overwritten.

4. The problems and our solutions

In this section we describe the various problems that we have encountered in developing Ashtex. The main problem was one of speed because a tool such as a previewer has to be fast if it is going to be used at all. The second one was a problem of input management because using mouse input is rather difficult; the programs quickly become complex and hard to maintain. The third problem was a fonts problem because of the unusual resolution of the Numelec bitmap.

4.1. Speed

The previewer has to run quickly in two ways: first the modifications to the screen such as moving a window, scrolling, or popping a window have to be done just when they are requested by the user. Likewise ghosts and thumbs must move smoothly. Pages should not seem to be built from small rectangles. Secondly the pages have to be computed by the driver quickly enough, so that the user doesn't wait too long when asking for a new page.

The first point is handled through ASH which is a very powerful screen handler. It uses a lot of memory (keeping a copy of each virtual bitmap) but makes it possible to modify the screen very quickly. For the second point we had to improve the performance of the driver, particularly the part of the driver computing the pages. To start with, the driver was only used for outputting on the printer, where speed wasn't so important. But when the user asks for a new page in an interactive previewer, it is not possible to let him wait more than a few seconds before the page appears on the screen. By modifying crucial parts of the code the computation time of a page on the SM90 was decreased from more than 14 seconds to approximately 4 seconds. On a more powerful workstation, incorporating a 68020 rather than a 68010 microprocessor, it is clear that this time will drop below 1 second. This work was performed by J. Incerpi and F. Montagnac and will be reported elsewhere.

4.2. Input management

To resolve the problem of input management in a graphical user interface, some languages exist that can help the programmer to describe interactions using a mouse and that generate C code which can be integrated in the graphical application. The language that we have chosen to help us for managing input in Ashtex, is a general synchronous programming language named *ESTEREL*, developed by G. Berry [2,3] at CMA in Sophia Antipolis. We haven't used ESTEREL for the entire input management of Ashtex, but for a complex part of it: the scrollbars. Berry has described the behavior of a scrollbar in ESTEREL and his system has generated a C automaton that we have integrated in Ashtex. This method has been useful in many ways. First of all the C code generated is compact and optimized. There is only one ESTEREL program for both the vertical and horizontal scrollbars. This method also provides modularity: the input management is completely separated from the rest of the program. And finally, to modify the behavior of the scrollbar, it is much easier to modify the ESTEREL program that more closely reflects the real behavior than to modify a big automaton implemented in a conventional language.

4.3. Missing Fonts

Using Ashtex, we have quickly encountered the problem of missing fonts. For the resolution corresponding to the Numelec bitmap (around 100 dots per inch) we have a library of 118 dots per inch fonts but it is not very complete. This library contains all the fonts for TEX without magnification but if a document has a magnification command then some fonts may be missing

and the driver will not be able to compute some pages of the document. If the magnification commands are removed from the TEX source file, the document can be displayed on the screen through Ashtex. However this is not a satisfactory solution because removing the magnification commands from the TEX file changes the layout of the paragraphs and pages. The output of the document in Ashtex will be very different than the output on paper. Not to mention that the document would have to be run through TEX another time to put back in the magnification desired before outputting on paper. Clearly having a full set of fonts for the device is the best solution but barring this we would like to see output that is as close to that on paper as possible. Retexing the file with and without magnification is not the solution. Thus our previewer offers an option to take the dvi file generated from a TEX source file with some overall magnification, and display it using the default (non-magnified) fonts. This works in general since if a font exists at all it is in this non-magnified size. This seems to be a reasonable solution since the page layout and line breaks are those that will appear in the hardcopy version, it is just that the characters are smaller. Note the spacing within the lines is proportional to the fonts used so that proofreading and checking layout is not hindered.

5. Conclusion

At this time Ashtex runs and is used on SM90 workstation and Ridge computer. We are planning to move it to a Sun workstation in the near future. To move Ashtex to a different environment first requires that ASH be adapted to this new machine. We've already mentioned that ASH's design makes this rather easy to do. As well, the code for the mouse management, which is very machine dependent, would have to be adjusted. Ashtex is written, however, in a very modular way. Our previewer consists of modules for computing pages from the dvi file, for the scrollbars, for the graphical layout of the screen, and for handling keyboard and mouse input. This not only makes porting Ashtex to another machine easy but could allow using the previewer environment for manipulating different objects. That is, rather than displaying pages of TEX output, we could manipulate digital pictures using most of the components of our system. Many people have used Ashtex in a daily fashion to prepare documents. The user-community that helped in defining its functionalities seems satisfied with the current compromises. The performance is considered acceptable, although improvements in this area are always welcome. The cycle for writing documents typeset with TEX still has three separate steps: writing source file, then running TEX, and finally displaying the document. This is not entirely satisfactory. There is actually no interaction between the output on the screen and the source file. From the source file to the displayed output, one has to run TEX to get the dvi file and while previewing a part of the output (a word, a line, an equation, or a page) there is no way to get the corresponding string in the source file. Using an multi-window previewer such as Ashtex, the user ideally wants to pick something on the screen and from within another window modify the corresponding string and see the newly adjusted output without leaving Ashtex. We aren't as yet able to create such a powerful tool for the TEX user. Although we can see very well how desirable it would be.

Bibliography

1. *The Brown Workstation Environment Documentation,* Brown University, October 1984.
2. Gérard Berry, *Programming a Digital Watch in ESTEREL,* CMA ENSMP, Sophia-Antipolis, February 1986.
3. Georges Gonthier, *The ESTEREL v2.1 Language Reference Manual,* CMA ENSMP, Sophia-Antipolis, March 1986.

ANNEXE

Using Ashtex

To run Ashtex some shell variables must first be initialized. One is the YRPIXELS variable; this variable contains the pathname of the directory containing the TEX pixel files for the bitmap resolution. For example : YRPIXELS=/usr/local/tex/pxl118

To use Ashtex on SM90, ones also needs the NUFT shell variable, that contains the pathnames of the directories containing the Numelec fonts. These fonts are used by ASH for writing windows titles, menus buttons, and for the dialog with the user.

Ashtex is distribued with the TEX pixel fonts, and a script shell called "aztex", initializing the YRPIXELS and NUFT variables and calling Ashtex. The user can either call "aztex" (if the variables are not set) or "ashtex" .

Calling Ashtex:

ashtex [-f dvifile_name] [-p page_number] [-r range_number]

The arguments are optional. If an argument is missing, the program asks the user for it at the beginning of the session.

-f dvifile_name	Name of the dvifile to load. The ".dvi" suffix is optional.
-p page_number	Number of the page to display first.
-r range_number	To display the selected page from the selected range. Note that ranges are numbered starting at 1. In a document containing pages 1-10, followed by pages 1-5, specifying "-r 2" means a page of the second group will be displayed.

Examples:

1) ashtex -f a -p 2
Displays the second page of the document "a.dvi", if this document has only one range of page numbers, else the program asks for the desired range.

2) ashtex -f a.dvi
If the document has only one page, it displays this page, else it asks for the page (and the range) to display.

3) ashtex -f a.dvi -r 2 -p 3
Displays the third page of the second range of the "a.dvi" document.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique